Where Ignoring Delete Lists Works, Part II: Causal Graphs

Joerg Hoffmann

Where Ignoring Delete Lists Works, Part II: Causal Graphs

Jörg Hoffmann
INRIA
Nancy, France
joerg.hoffmann@inria.fr

Abstract

The ignoring delete lists relaxation is of paramount importance for both satisficing and optimal planning. In earlier work (Hoffmann 2005), it was observed that the optimal relaxation heuristic h^+ has amazing qualities in many classical planning benchmarks, in particular pertaining to the complete absence of local minima. The proofs of this are hand-made, raising the question whether such proofs can be lead automatically by domain analysis techniques. In contrast to earlier disappointing results (Hoffmann 2005) – the analysis method has exponential runtime and succeeds only in two extremely simple benchmark domains – we herein answer this question in the affirmative. We establish connections between causal graph structure and h^+ topology. This results in low-order polynomial time analysis methods, implemented in a tool we call TorchLight. Of the 12 domains where the absence of local minima has been proved, TorchLight gives strong success guarantees in 8 domains. Empirically, its analysis exhibits strong performance in a further 2 of these domains, plus in 4 more domains where local minima may exist but are rare. In this way, TorchLight can distinguish "easy" domains from "hard" ones. By summarizing structural reasons for analysis failure, TorchLight also provides diagnostic output indicating domain aspects that may cause local minima.

Introduction

The ignoring delete lists relaxation is of paramount importance for both satisficing and optimal planning (e.g., Bonet and Geffner 2001; Hoffmann and Nebel 2001; Richter, Helmert, and Westphal 2008; Helmert and Domshlak 2009). The planners based on it approximate, in a variety of ways, the optimal relaxation heuristic h^+ which itself is **NP**-hard to compute. As earlier observed (Hoffmann 2005), h^+ has some rather amazing qualities in many classical planning benchmarks. Figure 1 gives an overview of these results.[1]

The results divide domains into classes along two dimensions. We will herein ignore the horizontal dimension, which pertains to dead ends (easy-to-test powerful criteria implying that a task is "undirected"/"harmless" are known). The vertical dimension divides the domains into three classes, with respect to the behavior of exit distance, defined as $d-1$ where d is the distance to a state with strictly smaller h^+ value. In the "easiest" bottom class, there exist constant upper bounds on exit distance from both, states on local minima and states on benches (flat regions). In the figure, the bounds are given in square brackets. For example, in Logistics, the bound for local minima is 0 – meaning that no local minima exist at all – and the bound for benches is 1. In the middle class, a bound exists only for local minima; that bound is 0 (no local minima at all) for all domains shown. In the "hardest" top class, both local minima and benches may take arbitrarily many steps to escape.

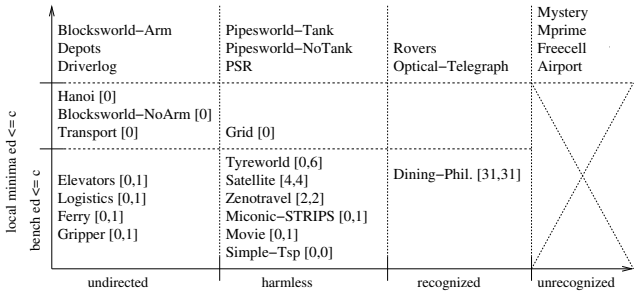

Figure 1: Overview of h^+ topology (Hoffmann 2005).

The proofs underlying Figure 1 are hand-made. For dealing with unseen domains, the question arises whether one can design domain analysis methods leading such proofs automatically. The potential uses of such analysis methods are manifold; we discuss this at the end of the paper. For now, note that addressing this question is quite a formidable challenge – at least if we are willing to take as an indication that (the decision problem is hard and) research trying to address it (Hoffmann 2005) managed only to design an analysis whose runtime explodes quickly with task size, and that succeeds only in Movie and Simple-TSP – arguably the two most simplistic benchmarks in existence.[2]

By contrast, the TorchLight tool developed herein has low-order polynomial runtime and usually terminates in split seconds. Distinguishing between *global* (per task) and *lo-*

[1] We omit ADL domains, and we add the more recent IPC benchmarks Elevators and Transport (without action costs), for which these properties are trivial to prove based on the earlier results. Blocksworld-Arm is the classical blocksworld, Blocksworld-NoArm is a variant allowing to "move A from B to C" directly.

[2] Simple-TSP encodes TSP but on a fully connected graph with uniform edge cost. The domain was originally introduced as a benchmark for symmetry detection.

cal (per state) analysis, it proves the global absence of local minima in Movie, Simple-TSP, Logistics, and Miconic-STRIPS. It gives a strong guarantee for local analysis in Ferry, Gripper, Elevators, and Transport. Empirically its local analysis exhibits strong performance also in Zenotravel, Satellite, Tyreworld, Grid, Driverlog, and Rovers. In this way, TorchLight can distinguish domains with "easy" h^+ topology from "hard" ones, even based on analyzing only a single state per instance. By summarizing structural reasons for analysis failure, TorchLight also gives diagnostic output indicating problematic aspects of the domain, i.e., aspects that may cause local minima under h^+.

What is the key to this performance boost? Consider Logistics and Blocksworld-Arm. At the level of their PDDL domain descriptions, the difference is not evident – both have delete effects, so why do those in Blocksworld-Arm "hurt" and those in Logistics don't? What does the trick is to move to the finite-domain variable representation (e.g., Helmert 2006) and to consider the associated structures, notably the causal graph capturing the precondition and effect dependencies between variables. The causal graph of Blocksworld-Arm contains cycles. That of Logistics does not. Looking into this, it was surprisingly easy to derive the following basic result:

If the causal graph is acyclic, and every variable transition is invertible, then there are no local minima under h^+.

This result is certainly interesting in that, for the first time, it establishes a connection between causal graph structure and h^+ topology. However, by itself the result is much too weak for domain analysis – of the considered benchmarks, it applies only in Logistics. We devise generalizations and approximations yielding the analysis results described above. Aside from their significance for domain analysis, these techniques are also interesting with respect to research on causal graphs. Whereas traditional methods (e.g., Jonsson and Bäckström 1995; Brafman and Domshlak 2004) seek execution paths solving the overall task, here we seek "only" execution paths decreasing the value of h^+. In local analysis, this enables us to consider small fragments of the causal graph, creating the potential to successfully analyze states in tasks whose causal graphs are otherwise arbitrarily complex.

We next give some background, then describe an illustrative example before delving into the technicalities. After pointing out some domain-specific performance guarantees, we report on a large-scale experiment with TorchLight. We close by discussing related and future work. We omit many details, and only outline the proofs. Full details are available in the journal version of the paper (Hoffmann 2011).

Background

We adopt the terminology and notation of Helmert (2006), with a number of modifications suiting our purposes. A (finite-domain variable) *planning task* is a 4-tuple (X, s_I, s_G, O). X is a finite set of *variables*, where each $x \in X$ is associated with a finite domain D_x. A *partial state* over X is a function s on a subset X_s of X, so that $s(x) \in D_x$ for all $x \in X_s$; s is a *state* if $X_s = X$. The *initial state* s_I is a state. The *goal* s_G is a partial state. O is a finite set of *operators*. Each $o \in O$ is a pair $o = (\mathrm{pre}_o, \mathrm{eff}_o)$ of partial states, called its *precondition* and *effect*. The semantics of planning tasks are defined as usual, based on allowing to transition from s to s' via o if $\mathrm{pre}_o \subseteq s$, $\mathrm{eff}_o \subseteq s'$, and $s(x) = s'(x)$ for all $x \in X \setminus X_{\mathrm{eff}_o}$. We denote the state space with S. We identify partial states with sets of variable-value pairs, which we will often refer to as *facts*.

We next define the two basic structures in our analysis: domain transition graphs and causal graphs. For the former, we diverge from Helmert's definition only in introducing additional notations for responsible operators and "side effects". In detail, let $x \in X$. The *domain transition graph* DTG_x of x is the labeled directed graph with vertex set D_x and the following arcs. For each $o \in O$ where $x \in X_{\mathrm{pre}_o} \cap X_{\mathrm{eff}_o}$ with $c := \mathrm{pre}_o(x)$ and $c' := \mathrm{eff}_o(x)$, DTG_x contains an arc (c, c') labeled with *responsible operator* $\mathrm{rop}(c, c') := o$, with *conditions* $\mathrm{cond}(c, c') := \mathrm{pre}_o \setminus \{(x, c)\}$, and with *side effects* $\mathrm{seff}(c, c') := \mathrm{eff}_o \setminus \{(x, c')\}$. For each $o \in O$ where $x \in X_{\mathrm{eff}_o} \setminus X_{\mathrm{pre}_o}$ with $c' := \mathrm{eff}_o(x)$, for every $c \in D_x$ with $c \neq c'$, DTG_x contains an arc (c, c') labeled with $\mathrm{rop}(c, c') := o$, $\mathrm{cond}(c, c') := \mathrm{pre}_o$, and $\mathrm{seff}(c, c') := \mathrm{eff}_o \setminus \{(x, c')\}$.

The reader familiar with causal graphs may have wondered why we introduced a notion of side effects, seeing as causal graphs can be acyclic only if all operators are unary (affect only a single variable). The reason is that we do handle cases where operators are non-unary. The variant of causal graphs we use can still be acyclic in such cases, and indeed this happens in some of our benchmark domains. We define the *support graph* SG to be the directed graph with vertex set X, and with an arc (x, y) iff DTG_y has a relevant transition (c, c') so that $x \in X_{\mathrm{cond}(c,c')}$. Here, a transition (c, c') on variable x is called *relevant* iff $(x, c') \in s_G \cup \bigcup_{o \in O} \mathrm{pre}_o$.

Our definition modifies the most commonly used one in that it uses relevant transitions only, and that it does not introduce arcs between variables co-occurring in the same operator effect. Transitions with side effects are handled separately in our analysis. Note that irrelevant transitions occur naturally, in domains with non-unary operators. For example, unstacking a block induces the irrelevant transition making the arm non-empty, and departing a passenger in Miconic-STRIPS makes the passenger "not-boarded".

We now introduce the relevant notations pertaining to h^+ and its topology. Let $s \in S$ be a state. A *relaxed plan* for s is an operator sequence achieving s_G when allowing to transition from fact set s to fact set s' via o if $\mathrm{pre}_o \subseteq s$ and $s' = s \cup \mathrm{eff}_o$. $h^+(s)$ is the length of an optimal (shortest) relaxed plan for s, or $h^+(s) = \infty$ if no relaxed plan exists. Say that $0 < h^+(s) < \infty$. Then an *exit* is a state s' reachable from s so that $h^+(s') = h^+(s)$ and there exists a neighbor s'' of s' so that $h^+(s'') < h^+(s')$. The *exit distance* $ed(s)$ of s is the length of a shortest path to an exit, or $ed(s) = \infty$ if no exit exists. A path in S is called *monotone* iff there exist no two consecutive states s_1 and s_2 on it so that $h^+(s_1) < h^+(s_2)$. We say that s is a *local minimum* if there exists no monotone path to an exit.

The topology definitions, adapted from (Hoffmann 2005),

should be self-explanatory. (They are specific to h^+ only because we will not consider any other heuristics.) Regarding h^+, Helmert (2006) first described the presented adaptation to finite-domain variables. To understand the definition, just note that "ignoring deletes" essentially means to act as if "what was true once will remain true forever". This is exactly what we do to the finite-domain variables here.

Domain analysis has a long tradition in planning (e.g., Fox and Long 1998; Gerevini and Schubert 1998; Edelkamp and Helmert 1999; Rintanen 2000). However, there exists no prior work at all trying to automatically infer topological properties of a heuristic function – the single exception being the aforementioned disappointing results (Hoffmann 2005).[3] It is worth noting that such analysis is computationally hard:

Theorem 1 *It is* **PSPACE***-complete to decide whether or not the state space of a given planning task contains a local minimum, and given an integer K it is* **PSPACE***-complete to decide whether or not for all states s we have $ed(s) \leq K$. Further, it is* **PSPACE***-complete to decide whether or not a given state s is a local minimum, and given an integer K it is* **PSPACE***-complete to decide whether or not $ed(s) \leq K$.*

These results are hardly surprising, but have not been stated anywhere yet. The hardness results still hold when restricting the input to solvable tasks/states. Their proofs reduce plan existence, the trick being to flatten h^+ using a new operator that can always achieve the goal but that has a fatal side effect, and giving the planner a choice whether to solve this modified task or a custom-designed alternative one. In practice, computational hardness here is particularly challenging because, in most applications of domain analysis, we are not willing to run a worst-case exponential search. After all, the analysis will not actually solve the problem.

The reader will have noticed the state-specific analysis problems in Theorem 1. We distinguish between *global* analysis per-task, and *local* analysis per-state. Domain analysis traditionally considers only the global variant (or even more generalizing variants looking at only the PDDL domain file). While global once-and-for-all analysis is also the "holy grail" in the present work, local analysis has its advantages. It applies in any domain, including those that do contain local minima, or that don't but where global analysis is not strong enough to recognize this. Indeed, we will see that local analysis, based on very limited sampling, can be used to produce accurate information about a domain.

An Illustrative Example

The basic connection we identify between causal graphs and h^+ topology is quite simple. It is instructive to understand this first, before delving into the full results. Figure 2 shows fragments of the domain transition graphs (DTGs) of three variables x_0, x_1, and x_2. All DTG transitions here are assumed to be invertible, and to have no side effects.

[3]This method enumerates all ways in which facts may support each other in a non-redundant relaxed plan. If there is no "conflict" then h^+ is the *exact* solution distance. This applies only in Simple-TSP. A slightly more general special case applies in Movie and trivial Logistics tasks with 2 locations, 1 truck, and 1 package.

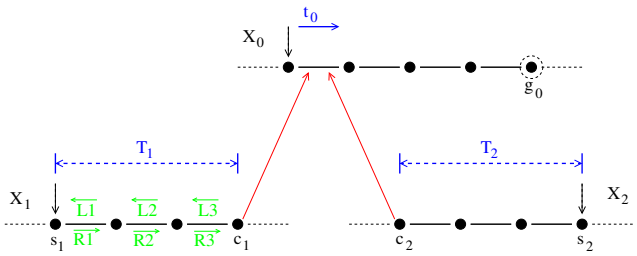

Figure 2: An example illustrating our basic result.

The imaginative reader is invited to think of x_0 as a car whose battery is currently empty and that therefore requires the help of two people, x_1 and x_2, in order to push-start it. The people may, to solve different parts of the task, be required for other purposes too, but here we consider only the sub-problem of achieving the goal $x_0 = g_0$. We wish to take the x_0 transition t_0, which has the two conditions c_1 and c_2. These conditions are currently not fulfilled. In the state s at hand, x_1 is in s_1 and x_2 is in s_2. We must move to a different state, s_0, in which $x_1 = c_1$ and $x_2 = c_2$. What will happen to h^+ along the way?

Say that an optimal relaxed plan $P^+(s)$ for s moves x_1 to c_1 along the path marked T_1, and moves x_2 to c_2 along the path marked T_2 – clearly, some such paths will have to be taken by any $P^+(s)$. Key observation 1 is similar to a phenomenon known from transportation benchmarks. When moving x_1 and x_2, whichever state s' we are in, as long as s' remains within the boundaries of the values traversed by T_1 and T_2, we can construct a relaxed plan $P^+(s')$ for s' so that $|P^+(s')| \leq |P^+(s)|$. Namely, to obtain $P^+(s')$, we simply replace the respective move sequence $\overrightarrow{o}_i$ in $P^+(s)$, for $i = 1, 2$, with its inverse $\overleftarrow{o}_i$. For example, say we got to s' by $\overrightarrow{o}_1 = \langle R1, R2, R3 \rangle$ moving x_1 to c_1, as indicated in Figure 2. Then wlog $P^+(s)$ has the form $\langle R1, R2, R3 \rangle \circ P$. We define $P^+(s') := \langle L3, L2, L1 \rangle \circ P$. The postfix P of both relaxed plans is the same; at the end of the prefix, the set of values achieved for x_1, namely s_1, c_1, and the two values in between, is also the same. Thus $P^+(s')$ is a relaxed plan for s'. This is true in general, i.e., $\overleftarrow{o}_1$ is necessarily applicable in s', and will achieve, within relaxed execution of $P^+(s')$, the same set of facts as achieved by $\overrightarrow{o}_1$ in $P^+(s)$. Thus $h^+(s') \leq h^+(s)$ for any state s', including the state s_0 we're after. Key observation 2: if x_0 moves only for its own sake, i.e., the car position is not important for any other goal, then h^+ will decrease by 1 after executing t_0 in s_0. Thus s_0 is an exit. With observation 1, the path to s_0 is monotone, hence s is not a local minimum, and its exit distance is bounded by the number of moves on the path.

It is not difficult to imagine that the above works also if preconditions need to be established recursively, as long as no cyclic dependencies exist. A third person may be needed to first persuade x_1 and x_2, the third person may need to take a bus, and so on. Now, say that the support graph is acyclic, and that all transitions are invertible and have no side effects. Given any state s, unless s is already a goal state, some variable x_0 moving only for its own sake necessarily exists. But then, within any optimal relaxed plan for s, a situation as above exists, and therefore we have a monotone exit path, *Q.E.D. for no local minima under h^+.*

The execution path construction we have just discussed is not so different from known results exploiting causal graph acyclicity and notions of connectedness or invertibility of domain transition graphs, e.g., (Jonsson and Bäckström 1995). What is new here is the connection to h^+.

The technical results in what follows are structured in a way similar to the above: (A) we identify circumstances under which we can deduce from an optimal relaxed plan that a monotone exit path exists. (B) we devise support-graph based sufficient criteria implying that analysis (A) will always succeed. (B) underlies TorchLight's global analysis. By feeding (A) with the usual relaxed plans as computed, e.g., by FF's heuristic function, we obtain TorchLight's main instrument for (approximate) local analysis. Thus we do not provide only a minimal version of (A) that would suffice to prove (B). We identify special cases much richer than what we can actually infer from support graphs (as yet).

Analyzing Optimal Relaxed Plans

We consider a state s and an optimal relaxed plan $P^+(s)$ for s. To describe the circumstances under which a monotone exit path is guaranteed to exist, we will need a number of notations pertaining to properties of transitions etc. We will introduce these notations along the way, rather than up front, in the hope that this makes them easier to digest.

Given $o_0 \in P^+(s)$, by $P^+_{<0}(s)$ and $P^+_{>0}(s)$ we denote the parts of $P^+(s)$ in front of o_0 and behind o_0, respectively. By $P^+(s,x)$ we denote the sub-sequence of $P^+(s)$ affecting x. We capture the dependencies between the variables used in $P^+(s)$ for achieving the precondition of o_0, as follows:

Definition 1 *Let (X, s_I, s_G, O) be a planning task, let $s \in S$ with $0 < h^+(s) < \infty$, let $P^+(s)$ be an optimal relaxed plan for s, let $x_0 \in X$, and let $o_0 \in P^+(s)$ be an operator taking a relevant transition of the form $t_0 = (s(x_0), c)$.*

An optimal rplan dependency graph *is a graph $oDG^+ = (V, A)$ with unique leaf vertex x_0, and where $x \in V$ and $(x, x') \in A$ if either: $x' = x_0$, $x \in X_{\mathrm{pre}_{o_0}}$, and $\mathrm{pre}_{o_0}(x) \neq s(x)$; or $x \neq x' \in V \setminus \{x_0\}$ and there exists $o \in P^+_{<0}(s)$ taking a relevant transition on x' so that $x \in X_{\mathrm{pre}_o}$ and $\mathrm{pre}_o(x) \neq s(x)$.*

For $x \in V \setminus \{x_0\}$, by $oDTG^+_x$ we denote the sub-graph of DTG_x that includes only the values true at some point in $P^+_{<0}(s,x)$, the relevant transitions t using an operator in $P^+_{<0}(s,x)$, and at least one inverse of such t where an inverse exists. We refer to the inverse transitions as induced*.*

The transition t_0 with responsible operator o_0 will be our candidate for reaching the exit state, like t_0 in Figure 2. oDG^+ collects all variables x connected to a variable x' insofar as $P^+_{<0}(s)$ uses an operator preconditioned on x in order to move x'. These are the variables we will need to move, like x_1 and x_2 in Figure 2, to obtain a state s_0 where t_0 can be taken. For any such variable x, $oDTG^+_x$ captures the domain transition graph fragment that $P^+_{<0}(s)$ traverses and within which we will stay, like T_1 and T_2 in Figure 2.

Note that there is no need to consider the operators $P^+_{>0}(s)$ behind o_0, simply because these operators are not used in order to establish o_0's precondition. This is of paramount importance in practice. For example, if o_0 picks up a ball b in Gripper, then $P^+(s)$ will also contain – behind o_0 – an operator o' dropping b. If we considered o' in Definition 1, then oDG^+ would contain a cycle because the definition would assume that o' is used for making the respective gripper hand free. In TorchLight's approximate local analysis, whenever we consider an operator o_0, before we build oDG^+ we re-order $P^+(s)$ by moving operators behind o_0 if possible. This minimizes $P^+_{<0}(s)$, and oDG^+ thus indeed contains only the necessary variables and arcs.

Under which circumstances will t_0 actually "do the job"? The sufficient criterion we identify is rather complex. To provide an overview of the criterion, we next state its definition. The items in this definition will be explained below.

Definition 2 *Let (X, s_I, s_G, O), s, $P^+(s)$, x_0, o_0, t_0, and $oDG^+ = (V, A)$ be as in Definition 1. We say that oDG^+ is* successful *if all of the following holds:*

(1) oDG^+ is acyclic.

(2) We have that either:
- *(a) the oDG^+-relevant deletes of t_0 are $P^+_{>0}(s)$-recoverable; or*
- *(b) $s(x_0)$ is not oDG^+-relevant, and t_0 has replacable side effect deletes; or*
- *(c) $s(x_0)$ is not oDG^+-relevant, and t_0 has recoverable side effect deletes.*

(3) For $x \in V \setminus \{x_0\}$, all $oDTG^+_x$ transitions either have self-irrelevant deletes, or are invertible/induced and have irrelevant side effect deletes and no side effects on $V \setminus \{x_0\}$.

The most basic prerequisite, (1), is that oDG^+ is acyclic. If there are cycles, then moving a variable may involve moving itself in the first place, which of course kills our exit path construction. For the $oDTG^+_x$ transitions, we are fine if, as in the example, all transitions are invertible and have no side effects. However, we can easily generalize this condition, to the present condition (3). We need some notations. Let $t = (c, c')$ be a transition on variable x. The *context* of t is the set $\mathrm{ctx}(t)$ of all facts that may be deleted by side effects of t. That is, for each $(y, d) \in \mathrm{seff}(t)$, $(y, \mathrm{cond}(t)(y)) \in \mathrm{ctx}(t)$ if a condition on y is defined; else all D_y values $\neq d$ are inserted. We say that:

- t is *invertible* iff there exists a transition (c', c) in DTG_x so that $\mathrm{cond}(c', c) \subseteq \mathrm{cond}(c, c')$.
- t has *irrelevant side effect deletes* iff $\mathrm{ctx}(t) \cap (s_G \cup \bigcup_{o \in O} \mathrm{pre}_o) = \emptyset$, and *self-irrelevant side effect deletes* iff $\mathrm{ctx}(t) \cap (s_G \cup \bigcup_{\mathrm{rop}(t) \neq o \in O} \mathrm{pre}_o) = \emptyset$.
- t has *self-irrelevant deletes* iff it has self-irrelevant side effect deletes and $(x, c) \notin s_G \cup \bigcup_{\mathrm{rop}(t) \neq o \in O} \mathrm{pre}_o$.

A transition is invertible if we can "go back" without introducing any new conditions (e.g. Logistics). Examples of irrelevant side effect deletes are transitions with no side effects at all, or a move in Simple-TSP, whose side effect deletes the target location's being "not-visited". Examples of self-irrelevant deletes are inflating a spare wheel in Tyreworld (the wheel is no longer "not-inflated"), or departing a passenger in Miconic-STRIPS, whose own effect deletes "not-served(passenger)" and whose side effect delete "boarded(passenger)" is used only by this operator itself.

How do these notions affect our exit path construction? Recall that we are trying to traverse states s' within the value ranges for $x \in V \setminus \{x_0\}$ defined by $oDTG^+_x$, to reach a state s_0 where we can apply t_0. Along the way, we construct relaxed plans $P^+(s')$ with $|P^+(s')| \leq |P^+(s)|$ by inverting transitions of $P^+_{<0}(s)$. If all $oDTG^+_x$ transitions t we may be using have irrelevant side effect deletes, then, as far as not disvalidating any facts needed elsewhere is concerned, this is just as good as having no side effects at all.[4] Regarding the own delete of t, this poses no problem here if t is invertible. If it is not but all deletes of t are irrelevant except maybe for the responsible operator itself, then to obtain $P^+(s')$ we can simply remove $\mathrm{rop}(t)$ from $P^+(s)$. Thus $|P^+(s')| < |P^+(s)|$ so we have reached an exit and can stop.

Consider now our endpoint transition t_0 and its responsible operator o_0. We previously demanded that x_0 "moves for its own sake", i.e., that x_0 has a goal value and is not important for achieving any other goal. This is unnecessarily restrictive. For example, in Driverlog, a driver may have its own goal *and* be needed to drive vehicles, and still t_0 moving the driver results in decreased h^+ if the location moved away from is not actually needed anymore. All we really want is that "any deletes of t_0 are not needed in the rest of the relaxed plan". We can then remove o_0 from the relaxed plan, and have reached an exit as desired.

Recall the situation we are addressing. We have reached a state s_0 in which $t_0 = (s(x_0), c)$ can be applied, yielding a state s_1. We have a relaxed plan $P^+(s_0)$ for s_0 so that $|P^+(s_0)| \leq |P^+(s)|$, where $P^+(s_0)$ is constructed from $P^+(s)$ by replacing some operators of $P^+_{<0}(s)$ with operators responsible for induced $oDTG^+_x$ transitions for $x \in V \setminus \{x_0\}$. We construct P^+_1 by removing o_0 from $P^+(s_0)$, and we need P^+_1 to be a relaxed plan for s_1. What are the facts possibly needed in P^+_1? A safe approximation is the union of s_G, the precondition of any $o_0 \neq o \in P^+(s)$, and any $oDTG^+_x$ values needed by induced $oDTG^+_x$ transitions. Denote that set with R^+_1. The values potentially deleted by t_0 are contained in $C_0 := \{(x_0, s(x_0))\} \cup \mathrm{ctx}(t_0)$. Thus if $R^+_1 \cap C_0 = \emptyset$ then we are fine. We can sharpen this further. Consider the set of facts F_0 that are true after relaxed execution of $P^+_{<0}(s)$. If $p \notin F_0$, then p is not needed in the part of P^+_1 pertaining to $P^+_{<0}(s)$; if it is needed in the part pertaining to $P^+_{>0}(s)$, behind o_0, then it is established by some operator $o' \in P^+_{>0}(s)$. Since o' will still be in P^+_1, it thus suffices if $R^+_1 \cap C_0 \cap F_0 = \emptyset$. Now, even this last condition can still be sharpened. Say that there exists a (possibly empty) sub-sequence $\overrightarrow{o_0}$ of $P^+_{>0}(s)$ so that $\overrightarrow{o_0}$ is applicable at the start of P^+_1, and $\overrightarrow{o_0}$ re-achieves all facts in $R^+_1 \cap C_0 \cap F_0$ (both are easy to define and test). Then moving $\overrightarrow{o_0}$ to the start of P^+_1 does the job. We say in this case that *the oDG^+-relevant deletes of t_0 are $P^+_{>0}(s)$-recoverable.* For example, say o_0 picks up ball b in Gripper. Then $P^+_{>0}(s)$ contains a sub-sequence $\overrightarrow{o_0}$ moving to another room and dropping b. This re-achieves $R^+_1 \cap C_0 \cap F_0 =$ "free-gripper".

[4] We must require no side effects on $V \setminus \{x_0\}$ due to a subtlety in the construction of $P^+(s')$. We omit this for brevity.

Definition 2 (2b) and (2c) identify two alternative sufficient conditions under which t_0 is suitable. Both require that $s(x_0)$ is not contained in R^+_1. We say in this case that $s(x_0)$ *is not oDG^+-relevant.* Then, $R^+_1 \cap C_0 = \emptyset$ unless t_0 has side effects. *Replacable side effect deletes* means, in a nutshell, that any endangered operator can always be replaced with an equivalent one (this happens, e.g., in Simple-TSP). *Recoverable side effect deletes* means that there exists an operator o' that will be applicable and recovers all relevant side effect deletes (e.g., in Rovers taking a rock/soil sample fills a "store", but we can easily empty the store again). Reaching an exit then takes one more step, applying o' after o_0.

What will the length of the exit path be? We have one move for x_0. Each non-leaf variable x must provide a new value at most once for every move of a variable x' depending on it, i.e., where $(x, x') \in A$. The new value can be reached by a $oDTG^+_x$ traversal. Denote the maximum length of such a traversal by $\mathrm{d}(oDTG^+_x)$. Now, we may have $\mathrm{d}(oDTG^+_x) > \mathrm{d}(DTG_x)$ because $oDTG^+_x$ removes not only vertices but also arcs: there may be "short-cuts" not traversed by $P^+(s)$. Under certain circumstances it is safe to take these short-cuts. Say that *(*) all $oDTG^+_x$ transitions are invertible or induced, and all other transitions are either irrelevant, or have empty conditions and irrelevant side effect deletes.* While traversing a short-cut, it may happen that h^+ increases. But when we reach the end of the short-cut, we are back in the region of states s' where a relaxed plan $P^+(s')$ can be constructed as above. Denote by V^* the subset of $V \setminus \{x_0\}$ for which (*) holds. We define $\mathrm{cost}^{\mathrm{d}*}(oDG^+) := \sum_{x \in V} \mathrm{cost}^{\mathrm{d}*}(x)$, where $\mathrm{cost}^{\mathrm{d}*}(x) :=$

$$\begin{cases} 1 & x = x_0 \\ \mathrm{d}(oDTG^+_x) * \sum_{x':(x,x') \in A} \mathrm{cost}^{\mathrm{d}*}(x') & x \neq x_0, x \notin V^* \\ \min(\mathrm{d}(oDTG^+_x), \mathrm{d}(DTG_x)) * & x \neq x_0, x \in V^* \\ \sum_{x':(x,x') \in A} \mathrm{cost}^{\mathrm{d}*}(x') & \end{cases}$$

For Definition 2 (2a,b), the exit distance is $\mathrm{cost}^{\mathrm{d}*}(oDG^+) - 1$ because we count the step reducing h^+. Thus:

Theorem 2 *Let (X, s_I, s_G, O), s, $P^+(s)$, and oDG^+ be as in Definition 1. If oDG^+ is successful, then s is not a local minimum, and $ed(s) \leq \mathrm{cost}^{\mathrm{d}*}(oDG^+)$. If we have Definition 2 (2a) or (2b), then $ed(s) \leq \mathrm{cost}^{\mathrm{d}*}(oDG^+) - 1$.*

Theorem 2 does not hold if $P^+(s)$ is not optimal, even if $P^+(s)$ is non-redundant and parallel-optimal: at the end of the "exit path" we may obtain a relaxed plan shorter than $P^+(s)$ but not shorter than $h^+(s)$. My example proving this is fairly contrived, though, and it seems unlikely that situations like this will occur in practice.

Note that $\mathrm{cost}^{\mathrm{d}*}(.)$ is exponential in the depth of the graph. This is not an artifact of the length estimation. It is easy to construct examples where exit distance is exponential in that parameter.

Conservative Approximations

We now identify sufficient criteria guaranteeing that Theorem 2 can be applied. We consider both the local case where a particular state is given, and the global case where we generalize over all states in the task. We approximate optimal rplan dependency graphs as follows:

Definition 3 *Let (X, s_I, s_G, O) be a planning task, let $s \in S$ with $0 < h^+(s) < \infty$, let $x_0 \in X_{s_G}$, and let $t_0 = (s(x_0), c)$ be a relevant transition in DTG_{x_0} with $o_0 := \mathrm{rop}(t_0)$.*

A local dependency graph *is a graph $lDG = (V, A)$ with unique leaf vertex x_0, and where $x \in V$ and $(x, x') \in A$ if either: $x' = x_0$, $x \in X_{\mathrm{pre}_{o_0}}$, and $\mathrm{pre}_{o_0}(x) \neq s(x)$; or $x' \in V \setminus \{x_0\}$ and (x, x') is an arc in SG.*

A global dependency graph *is a graph $gDG = (V, A)$ with unique leaf vertex x_0, and where $x \in V$ and $(x, x') \in A$ if either: $x' = x_0$ and $x_0 \neq x \in X_{\mathrm{pre}_{o_0}}$; or $x' \in V \setminus \{x_0\}$ and (x, x') is an arc in SG.*

If an optimal relaxed plan $P^+(s)$ for s contains o_0, then oDG^+ as per Definition 1 will be a sub-graph of lDG and gDG as defined here. This is simply because any optimal rplan dependency graph has only arcs (x, x') contained in the support graph of the task.[5] I remark that the support graph may contain a lot more arcs than actually necessary. Consider the earlier point that, when constructing oDG^+, it is important to consider only operators *in front of* o_0 in $P^+(s)$. This information is of course not contained in SG. In Gripper, to stick with our previous example, SG will suggest that dropping a ball may be needed in order to support "free-gripper" for picking up the same ball. Thus SG "detects" a cyclic dependency here. One of the main open directions is to improve this part of the approximation, by devising conservative methods recognizing operators that will never have to precede o_0 in an optimal relaxed plan.

Defining when an lDG respectively gDG is successful does not involve any new notation:

Definition 4 *Let (X, s_I, s_G, O), s, x_0, t_0, o_0, and $G = lDG$ or $G = gDG$ be as in Definition 3. We say that $G = (V, A)$ is* successful *if all of the following hold:*

(1) G is acyclic.
(2) If $G = lDG$ then $s_G(x_0) \neq s(x_0)$, and there exists no transitive successor x' of x_0 in SG so that $x' \in X_{s_G}$ and $s_G(x') \neq s(x')$.
(3) We have that t_0 either:
  (a) has self-irrelevant side effect deletes; or
  (b) has replacable side effect deletes; or
  (c) has recoverable side effect deletes.
(4) For $x \in V \setminus \{x_0\}$, all DTG_x transitions either are irrelevant, or have self-irrelevant deletes, or are invertible and have irrelevant side effect deletes and no side effects on $V \setminus \{x_0\}$.

Consider first only local dependency graphs $G = lDG$; we will discuss $G = gDG$ below. Assume that we have an optimal relaxed plan $P^+(s)$ for s that contains o_0, and thus oDG^+ is a sub-graph of lDG. Then condition (1) obviously implies Definition 2 (1). Condition (4) implies Definition 2 (3) because $oDTG_x^+$ does not contain any irrelevant transitions. Condition (2) implies that $s(x_0)$ is not oDG^+-relevant, i.e., $s(x_0)$ is not needed in the rest of the relaxed plan. This is simply because no other un-achieved goal depends on x_0. But then, condition (3a) implies Definition 2 (2a) because $R_1^+ \cap C_0 = \emptyset$, in the notation introduced previously. Conditions (3b) and Definition 2 (2b), respectively (3c) and Definition 2 (2c), are equivalent under this premise.

Regarding exit distance, we do not know which part of $x \in V \setminus \{x_0\}$ will be traversed by $P^+(s)$. An obvious bound on $\mathrm{d}(oDTG_x^+)$ is the length $\mathrm{mD}(DTG_x)$ of a longest non-redundant path through the graph (a path visiting each vertex at most once). Unfortunately, we cannot compute $\mathrm{mD}(.)$ efficiently: there exists a Hamiltonian path in a graph $G = (V, A)$ iff $\mathrm{mD}(G) = |V| - 1$. TorchLight over-approximates $\mathrm{mD}(G)$ simply by $|V| - 1$. On a more positive note, we can sometimes use $\mathrm{d}(DTG_x)$ instead of $\mathrm{mD}(DTG_x)$, namely if we are certain that x is one of the variables V^* used in the definition of $\mathrm{cost}^{\mathrm{d}*}(oDG^+)$. This can be ensured by postulating that all DTG_x transitions either are irrelevant, or are invertible, have empty conditions, irrelevant side effect deletes, and no side effects on $V \setminus \{x_0\}$. Say that V^{**} contains this variable subset. We define $\mathrm{cost}^{\mathrm{D}*}(G) := \sum_{x \in V} \mathrm{cost}^{\mathrm{D}*}(x)$, where $\mathrm{cost}^{\mathrm{D}*}(x) :=$

$$\begin{cases} 1 & x = x_0 \\ \mathrm{mD}(DTG_x) * \sum_{x':(x,x') \in A} \mathrm{cost}^{\mathrm{D}*}(x') & x \neq x_0, x \notin V^{**} \\ \mathrm{d}(DTG_x) * \sum_{x':(x,x') \in A} \mathrm{cost}^{\mathrm{D}*}(x') & x \neq x_0, x \in V^{**} \end{cases}$$

Because x_0 must move – to attain its own goal – every optimal relaxed plan must take at least one transition leaving $s(x_0)$. Thus, with Theorem 2 and the above, we have that:

Theorem 3 *Let (X, s_I, s_G, O) be a planning task, and let $s \in S$ be a state with $0 < h^+(s) < \infty$. Say that $x_0 \in X$ so that, for every $o_0 = \mathrm{rop}(s(x_0), c)$ in DTG_{x_0} where $(s(x_0), c)$ is relevant, lDG_{o_0} is a successful local dependency graph. Then s is not a local minimum, and $ed(s) \leq \max_{o_0} \mathrm{cost}^{\mathrm{D}*}(lDG_{o_0})$. If, for every lDG_{o_0}, we have Definition 4 (3a) or (3b), then $ed(s) \leq \max_{o_0} \mathrm{cost}^{\mathrm{D}*}(lDG_{o_0}) - 1$.*

Theorem 3 is our tool for guaranteed local analysis, i.e., a search-state analysis that guarantees its information to be correct. For guaranteed global analysis, we simply look at the set of *all* global dependency graphs gDG, requiring them to be successful. In particular, all gDG are then acyclic, from which it is not difficult to deduce that any non-goal state s will have a variable x_0 fulfilling Definition 4 (2). For that x_0, we can apply Theorem 3 and thus get:

Theorem 4 *Let (X, s_I, s_G, O) be a planning task. Say that all global dependency graphs gDG are successful. Then S does not contain any local minima and, for any state $s \in S$ with $0 < h^+(s) < \infty$, $ed(s) \leq \max_{gDG} \mathrm{cost}^{\mathrm{D}*}(gDG)$. If, for every gDG, we have Definition 4 (3a) or (3b), then $ed(s) \leq \max_{gDG} \mathrm{cost}^{\mathrm{D}*}(gDG) - 1$.*

If SG is acyclic and all transitions are invertible and have no side effects, then Theorem 4 applies, whereby we have now in particular proved our basic result. Vice versa, note that, if Theorem 4 applies, then SG is acyclic. As far as local minima are concerned, one may thus reformulate Theorem 4 in simpler terms not relying on a notion of "successful dependency graphs". The present formulation already paves the way for future research: a gDG is defined relative to a concrete variable x_0 and operator o_0, and may thus allow for more accurate analysis of which other variables may actually become important for x_0 and o_0, in a relaxed plan.

[5] For gDG, note that $\mathrm{pre}_{o_0}(x_0)$, if defined, will be $= s(x_0)$ and thus x_0 does not need to be recorded as its own predecessor.

Benchmark Performance Guarantees

We now state some guarantees that our analysis gives in benchmark domains. The underlying finite-domain variable formalizations are straightforward.

Proposition 1 *Let (X, s_I, s_G, O) be a planning task from the Logistics, Miconic-STRIPS, Movie, or Simple-TSP domain. Then Theorem 4 applies, and the bound delivered is at most* 1*,* 3*,* 1*, and* 1 *respectively.*

Proposition 2 *Let (X, s_I, s_G, O) be a planning task from the Elevators, Ferry, Gripper, or Transport domain, and let $s \in S$. In Ferry and Gripper, for every optimal relaxed plan $P^+(s)$ there exists oDG^+ so that Theorem 2 applies, the bound being at most* 1*. In Elevators and Transport, there exists at least one $P^+(s)$ and oDG^+ so that Theorem 2 applies, the bound being at most* 1 *in Elevators and at most the road map diameter in Transport.*

Proposition 2 holds because all vehicle capacity deletes are recovered inside the relaxed plan. For Elevators and Transport, the result is slightly weaker because a vehicle may have capacity > 1, allowing relaxed plans to use unloading operators recovering a capacity not actually present.

Experiments

TorchLight is implemented in C based on FF-v2.3, using Fast-Downward's translator to find the finite-domain variables.[6] We run experiments in 37 domains, including those of Figure 1 and all IPC STRIPS domains up to 2008, except Cyber-Security which involved parsing and/or translation difficulties. The test instances, 1160 in total, were taken from the IPC collection(s) where applicable, and (randomly) generated elsewhere. All experiments are run on a 1.8 GHZ CPU, with a 30 minute runtime and 2 GB memory cut-off.

TorchLight runs global analysis checking the preconditions of Theorem 4. It then generates R sample states by random walks. We run $R = 1, 10, 100, 1000$. Each state is analyzed using guaranteed local analysis checking Theorem 3, as well as approximate local analysis checking Theorem 2 on a relaxed plan as computed by FF. The latter uses the aforementioned relaxed plan re-ordering technique, and a simple technique allowing to recognize situations where failure due to one operator can be avoided by replacing with an alternative operator. The local analyzes return statistics concerning in particular the *success rate*, i.e., the fraction of sample states where Theorem 3/Theorem 2 applied.

The code is currently optimized much more for readability than for speed. Still, TorchLight is fast. Up to $R = 100$, the main bottleneck is Fast-Downward's translator. With $R = 1$ and $R = 10$, in 99% of the instances the actual analysis takes at most as much time as the translator. With $R = 100$ this holds for 96%. On average, the translator takes 6.1 seconds; for analysis this is 0.8, 0.9, 1.9 with $R = 1, 10, 100$. For $R = 1000$, analysis can be more expensive, up to >1 minute in 5 domains. As we will see, this many samples are not needed to obtain useful information.

[6]The source code of TorchLight is available at `http://www.loria.fr/~hoffmanj/TorchLight.zip`

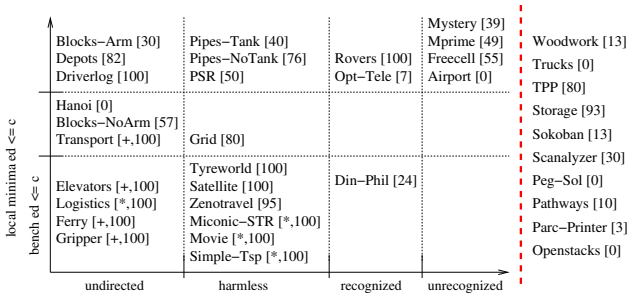

Figure 3: Overview of TorchLight domain analysis results. "*": global analysis always succeeds; "+": local analysis always succeeds if provided an optimal relaxed plan; mean success rates when sampling one state per domain instance.

The guarantees of Proposition 1 are confirmed, i.e., global analysis succeeds as described in Logistics, Miconic-STRIPS, Movie, and Simple-TSP. It never succeeds in any other domain, though. In some domains, fractions of the gDGs are successful, e.g., up to 97% in Satellite. However, if the fraction is below 100% then nothing is proved so this data may at best serve as an indication of which aspects of the domain are "good-natured". As for guaranteed local analysis, this generally is not much more applicable than global analysis. In what follows, we hence concentrate on approximate local analysis (via Theorem 2) exclusively.

Proposition 2 is backed up impressively. Approximate local analysis succeeds in every single sample state of Ferry, Gripper, Elevators, and Transport. Thus the potentially suboptimal relaxed plans do not hurt here. Indeed, the analysis yields high success rates in almost all domains where local minima are non-present or limited. This is not the case for the other domains, and thus TorchLight can distinguish domains with "easy" h^+ topology from "hard" ones. Consider Figure 3, showing mean success rates per-domain with $R = 1$. The picture is similar for $R = 10, 100, 1000$.

The domains whose h^+ topology is not known are shown separately on the right hand side in Figure 3. For the other domains, we see quite nicely that "harder" domains tend to have lower success rates. To some extent, we can even distinguish Pipesworld-Tankage from Pipesworld-NoTankage, and Mprime from Mystery (in Mprime, fuel can be transferred between locations). Strong outliers are Driverlog, Rovers, Hanoi, and Blocksworld-NoArm. But all of these are more problems of the hand-made analysis than of TorchLight's. In Driverlog and Rovers, deep local minima do exist, but only in awkward situations that don't tend to arise in the IPC instances. As for Hanoi and Blocksworld-NoArm, these are not actually easy to solve for FF, and the absence of local minima is due to rather idiosyncratic reasons.

Success rates can also be obtained by *search probing (SP)*. For each sample state, we search (using FF's helpful actions) for a monotone exit path under FF's heuristic. Runtime is worst-case exponential for any R, but in this experiment is competitive up to $R = 10$. For $R = 1000$, in 11 (4) domains runtime can be >1 (>30) minute(s). To some extent, one can handle this by a small time cut-off. Allowing 1 second per sample state changes the success rate by at most 5%, in 99% of these benchmarks. In general, though, of course such a cut-off severely limits the lookahead capability of search.

Remarkably, the prediction quality of TorchLight is just as good as – sometimes even better than – that of unlimited SP. For details, see (Hoffmann 2011). We consider primitive classifiers based on only the success rate, predicting whether or not FF's enforced hill-climbing search will succeed in solving the overall task. The classifiers answered "yes" iff the success rate was above a threshold T in $0, 10, \ldots, 100$. The best rate of correct predictions (T being around 80) is 71.8% ($R = 100$) and 71.9% ($R = 10$) for TorchLight vs. 71.2% and 70.2% for SP; a baseline always answering "yes" would achieve 60.7% success here. This simplistic prediction method does not work as well for FF as a whole and for LAMA, which solve many tasks with unfavorable h^+ topology. However, looking at the runtime distributions below and above T, the mean of the latter is significantly smaller in all cases for TorchLight, and in most cases for SP.

TorchLight can also tell us *why* analysis failed, identifying domain features causing the presence of local minima. Since the tested criteria are sufficient but not necessary, there is no correctness guarantee. Still, at least for analysis using Theorem 2, the diagnosis can be quite accurate. It currently does not do much more than report all variable/operator pairs failing to qualify for x_0, o_0. In Zenotravel, this always correctly identifies fuel consumption as the problem. In Mprime and Mystery, most of the time the same correct diagnosis is returned. In Satellite and Rovers, it always reports the problem to be that switching on an instrument, respectively taking an image, deletes calibration – precisely the only reason why local minima exist here. In Tyreworld, Grid, Blocksworld-Arm, and Freecell, the diagnosis identifies critical resources (like "hand-empty" and "have-cellspace"). It seems likely that these results could still be much improved, by combining them with other information sources in TorchLight and/or with other domain analysis techniques.

Discussion

We identified a connection between causal graphs and h^+, and devised a tool allowing to analyze search space topology without actually running any search. The tool is not yet an "automatic Hoffmann", but its analysis quality is impressive even when compared to unlimited search probing.

Causal graphs were previously used to identify tractable fragments (e.g., Jonsson and Bäckström 1995; Brafman and Domshlak 2004; Haslum 2007). However, tractability and absence of local minima are orthogonal properties. The strongest connection is that, if Theorem 4 applies, then plan existence (but not plan generation nor optimal planning) is tractable. For our basic result, but not for the full scope of Theorem 4, this tractability is known (Hoffmann 2011).

At a technical level, this work establishes two new aspects of causal graph analysis. (1) One can "localize" the analysis, considering only the fragment relevant for solving a particular state. This enables succes in tasks that are otherwise arbitrarily complex. (2) One can construct paths improving the value of a heuristic, rather than achieving the global goal. This is not limited to h^+. Both (1) and (2) suggest completely new avenues of causal graph research.

TorchLight is useful for performance prediction. Like limited SP, it generates a highly informative feature without jeopardizing runtime, thus enabling automatic planner configuration. Unlike for SP, this may even work on-line during search. Based on a single relaxed plan, e.g., one might choose a different search strategy, or switch helpful actions on/off, depending on the outcome of checking Theorem 2.

Local analysis can be used to generate macro-actions, following the path to the better state. This relates to work on macros (e.g., Botea, Müller, and Schaeffer 2004), but with a very targeted analytical way of generating them.

One could use TorchLight's diagnosis facility as the basis of an abstraction technique for deriving search guidance. The diagnosis can pin-point which operator effects are causing problems for search. If we remove enough harmful effects so that Theorem 4 applies, then the abstract problem is tractable. If we do not abstract that much, the information provided may still outweigh the effort for abstract planning.

Another interesting issue is domain reformulation. Haslum (2007) identifies as a main open problem the lack of guiding information. TorchLight's diagnosis may come to the rescue. Automatic reformulation could, e.g., precompose variable subsets touched by harmful effects, or hide these effects within macro-actions. The diagnosis could also give modeling advice to non-planning experts, supporting them in the design of hierarchies of domains giving different trade-offs between planning effort and plan accuracy, thus helping to make planning technology more accessible.

Acknowledgments. I had long ago given up on this problem. Thanks to Carlos Areces' and Luciana Benotti's insistence, I finally saw the connection to causal graphs – while trying to convince them that such an analysis is not possible.

References

Bonet, B., and Geffner, H. 2001. Planning as heuristic search. *AI*.

Botea, A.; Müller, M.; and Schaeffer, J. 2004. Using component abstraction for automatic generation of macro-actions. *ICAPS'04*.

Brafman, R., and Domshlak, C. 2004. Structure and complexity in planning with unary operators. *JAIR* 18:315–349.

Edelkamp, S., and Helmert, M. 1999. Exhibiting knowledge in planning problems to minimize state encoding length. *ECP'99*.

Fox, M., and Long, D. 1998. The automatic inference of state invariants in TIM. *JAIR* 9:367–421.

Gerevini, A., and Schubert, L. 1998. Inferring state-constraints for domain independent planning. *AAAI'98*, 905–912.

Haslum, P. 2007. Reducing accidental complexity in planning problems. *IJCAI'07*, 1898–1904.

Helmert, M., and Domshlak, C. 2009. Landmarks, critical paths and abstractions: What's the difference anyway? *ICAPS'09*.

Helmert, M. 2006. The fast downward planning system. *JAIR*.

Hoffmann, J., and Nebel, B. 2001. The FF planning system: Fast plan generation through heuristic search. *JAIR* 14:253–302.

Hoffmann, J. 2005. Where 'ignoring delete lists' works: Local search topology in planning benchmarks. *JAIR* 24:685–758.

Hoffmann, J. 2011. Analyzing Search Topology Without Running Any Search: On the Connection Between Causal Graphs and h^+. *JAIR*.

Jonsson, P., and Bäckström, C. 1995. Incremental planning. *European Workshop on Planning*.

Richter, S.; Helmert, M.; and Westphal, M. 2008. Landmarks revisited. *AAAI'08*, 975–982.

Rintanen, J. 2000. An iterative algorithm for synthesizing invariants. *AAAI'00*, 806–811.